NFA TO DFA CONVERSION

The python script, script.py, reads an input.json file and converts the NFA thus read in the file into a DFA.
The obtained DFA is then written into an output.json file.

Running the Script

The input.json file and the script.py file must be in the same directory. Simply run the script.py file and it will create an ouput.json file with the desired DFA output in the same directory.

Use the following command with python3 to run the script

```
python3 script.py
```

Brief description of an NFA and DFA

Consider an NFA

$N = < Q_n, q_n, \Sigma_n, \delta_n: Q_n \text{ x } \Sigma_n \text{ -->} Q_n, F_n >$

And its equivalent DFA

$D = < Q', q', \Sigma', \delta': Q' \text{ x } \Sigma' \text{ -->} Q', F' >$

Then the following relations exist:

- $Q' = 2^{Q_n}$ (Power set of Q_n)
- $q' = q_n$
- $\Sigma' = \Sigma_n$

Procedure of Conversion (CODE)

- Once the input file is read and the number of states in the NFA are known, the program creates the power set of Q_n

```
for c in range(0,pset_size):
    sub=[]
    for i in range(0,obj['states']):
        if c & 1<<i:
            sub.append(st[i])

    subsets.update({c:sub})
```

The collection of all subsets, that is the power set is stored as a json object, “subsets”. The subsets are formed as per this rule:
If there are “p” states in the NFA, then in the binary representation of each number of a state, namely $q_{n0}, q_{n1}, \ldots, q_{n(p-1)}$>, the bits that are high indicate those states that are a part of a particular state in the DFA.
For eg:
If there are 2 states in the NFA, then for q_3 binary representation is 11. Hence the state in the DFA will be $\{q_{n0}, q_{n1}\}$

- Next is the creation of the transition relation (or function) for the DFA. We begin by first including all the singleton set elements of the set Q’ and using the “t_func” of the NFA, we determine their relation in the DFA.

```
for i in range(1,max+1):
    for j in range(0,len(obj['letters'])):
        let=obj['letters'][j]
        a=func(let,stateno) #func returns the value on par
sing "tfunc" from the input file.
        relation.append([subsets[i],let,a])
    mark[i]=1 #used for other parts in the program.
    i=i*2
    stateno=stateno+1
```

Here the funtion func() takes in two parameters namely “let” and “stateno” where let is the letter and stateno is the state number. This loop runs over powers of 2, since in their binary representation, there is only one bit that is high thus indicating the subset of only a singleton element.

The function body of func() is as follows:

```
def func (a,stateno):
    arr=[]
    for i in range (0,len(obj['t_func'])):
        if(obj['t_func'][i][1]==a and obj['t_func'][i][0]=
=stateno):
            # arr.append(obj['t_func'][i][2])
            arr=obj['t_func'][i][2]
    return (arr)
```

This function parses through the “t_func” of the NFA json and returns the array of states that a particular state maps to on an input letter.

- Once all the singleton set elements are covered, the relations for the remaining elements of Q’ for each input alphabet of Σ can be found by simply taking the union of the results of their individual elements of the set.

- To determine the final states of the DFA, let R be a collection of states q.
  Then
  $F=\{R \in Q' \mid \exists\, q \in R \cap F_n \}$----- (1)

  Therefore first all the elements of F_n are added to the set of F. Following which all states in Q’ that satisfy conditon (1) are included (excluding repeat elements)

- Finally the output json is created, integrating all the elements of the DFA Q’. This json is then written into the output.json file.